

Computest

Hacking the pandemic's most popular software: Zoom



Zero Day Initiative

@thezdi

...

Confirmed! The duo of Daan Keuper and Thijs Alkemade from Computest used a 3-bug chain to exploit [#Zoom](#) messenger with 0 clicks from the target. They win \$200,000 and 20 points towards Master of Pwn. [#Pwn2Own](#)



Computest
always on.



<https://sector7.computest.nl/post/2021-08-zoom/>

About Pwn2Own

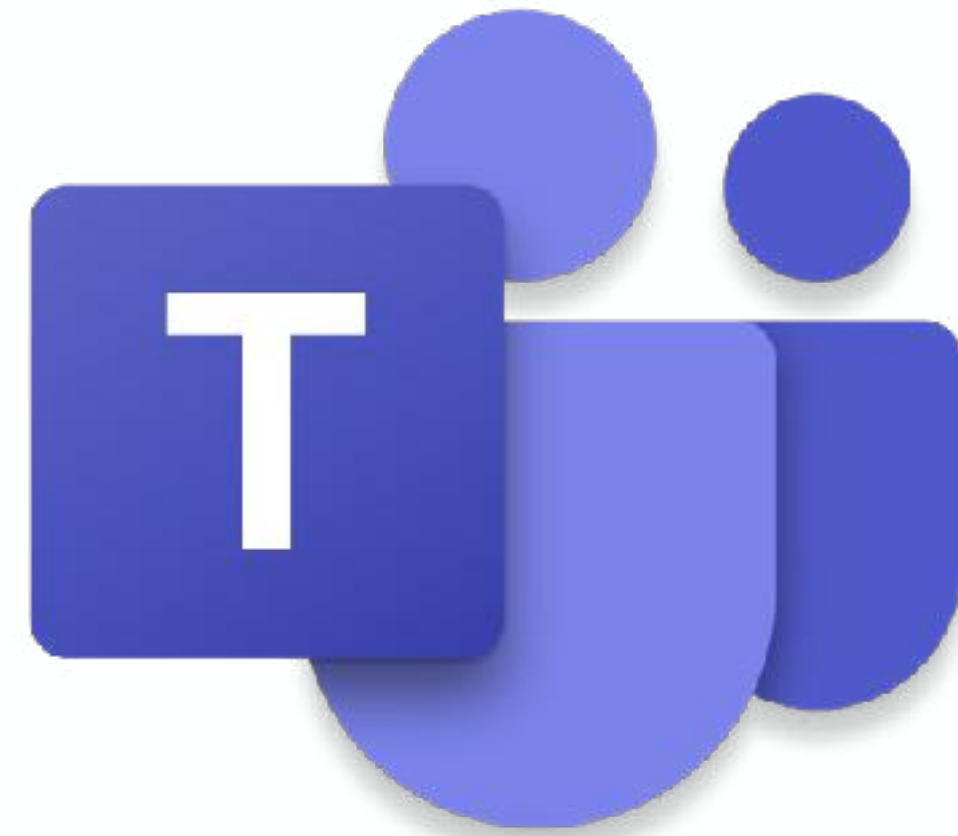
RO DAY
IATIVE
osoft
are®



Targets and Web Browsers

	Number of Pwn Points	Escape Options
Internet Explorer	1	1
Firefox	1	1
Chrome	1	1
Safari	1	1
Opera	1	1
Edge	1	1
Brave	1	1
Vivaldi	1	1
Chromium	1	1
WebKit	1	1
Gecko	1	1
Blink	1	1
Trident	1	1
EdgeHTML	1	1
WebKitGTK	1	1
WebKit2	1	1
WebKit3	1	1
WebKit4	1	1
WebKit5	1	1
WebKit6	1	1
WebKit7	1	1
WebKit8	1	1
WebKit9	1	1
WebKit10	1	1
WebKit11	1	1
WebKit12	1	1
WebKit13	1	1
WebKit14	1	1
WebKit15	1	1
WebKit16	1	1
WebKit17	1	1
WebKit18	1	1
WebKit19	1	1
WebKit20	1	1
WebKit21	1	1
WebKit22	1	1
WebKit23	1	1
WebKit24	1	1
WebKit25	1	1
WebKit26	1	1
WebKit27	1	1
WebKit28	1	1
WebKit29	1	1
WebKit30	1	1
WebKit31	1	1
WebKit32	1	1
WebKit33	1	1
WebKit34	1	1
WebKit35	1	1
WebKit36	1	1
WebKit37	1	1
WebKit38	1	1
WebKit39	1	1
WebKit40	1	1
WebKit41	1	1
WebKit42	1	1
WebKit43	1	1
WebKit44	1	1
WebKit45	1	1
WebKit46	1	1
WebKit47	1	1
WebKit48	1	1
WebKit49	1	1
WebKit50	1	1
WebKit51	1	1
WebKit52	1	1
WebKit53	1	1
WebKit54	1	1
WebKit55	1	1
WebKit56	1	1
WebKit57	1	1
WebKit58	1	1
WebKit59	1	1
WebKit60	1	1
WebKit61	1	1
WebKit62	1	1
WebKit63	1	1
WebKit64	1	1
WebKit65	1	1
WebKit66	1	1
WebKit67	1	1
WebKit68	1	1
WebKit69	1	1
WebKit70	1	1
WebKit71	1	1
WebKit72	1	1
WebKit73	1	1
WebKit74	1	1
WebKit75	1	1
WebKit76	1	1
WebKit77	1	1
WebKit78	1	1
WebKit79	1	1
WebKit80	1	1
WebKit81	1	1
WebKit82	1	1
WebKit83	1	1
WebKit84	1	1
WebKit85	1	1
WebKit86	1	1
WebKit87	1	1
WebKit88	1	1
WebKit89	1	1
WebKit90	1	1
WebKit91	1	1
WebKit92	1	1
WebKit93	1	1
WebKit94	1	1
WebKit95	1	1
WebKit96	1	1
WebKit97	1	1
WebKit98	1	1
WebKit99	1	1
WebKit100	1	1

Target	Escape Options	Prize	Master of Pwn Points
Google Chrome	Renderer Only	\$50,000	5
	Sandbox Escape	\$150,000	15
	Windows Kernel Escalation of Privilege	\$150,000	15
Microsoft Edge (Chromium)	Renderer Only	\$50,000	5
	Sandbox Escape	\$150,000	15
	Windows Kernel Escalation of Privilege	\$150,000	15
Apple Safari	Sandbox Escape	\$75,000	8
	macOS Kernel Escalation of Privilege	\$100,000	10
Mozilla Firefox	Sandbox Escape	\$50,000	5
	Windows Kernel Escalation of Privilege	\$60,000	6



Approach



Zoom

Vollet Davis

Sofi Kaiser

Mike Nolan

Josh Nelson

Madrid (5)

Q3 Outlook

Design functional products to help people get out and experience the world

OFFER NEW SERVICES

We're giving 1-month trials for consumers to test out our new products.

INCREASED SUPPLY

Raising our backstock 40% in Q3 to meet Q4 Demands

OFFER 24/7 SUPPORT

Headcount increased – 20 new openings

<

>

2 of 10

...

Mute

Stop Video

Security

Participants

Share Screen

Chat

Record

Reactions

End



The screenshot displays the Immunity Debugger interface with the following components:

- Program Tree:** Shows the loaded modules, including `packedup`.
- Symbol Tree:** Lists symbols for the `FUN_00400600` function.
- Data Type Manager:** Shows the data types for the `packedup` module.
- Listing: packedup:** Displays the unpacked ELF header for `FUN_00400600`. The header is an `ELF64_Ehdr` structure with the following fields:
 - `e_ident`: Magic number and other identification data.
 - `e_type`: `ET_EXEC` (Executable).
 - `e_machine`: `EM_X86_64` (AMD64 architecture).
 - `e_version`: `EV_CURRENT`.
 - `e_entry`: Entry point address.
 - `e_phoff`: Program header offset.
 - `e_shoff`: Section header offset.
 - `e_flags`: Processor-specific flags.
 - `e_ehsize`: Exception handling table size.
 - `e_phnum`: Number of program headers.
 - `e_phentsize`: Size of each program header entry.
 - `e_shnum`: Number of section headers.
 - `e_shstrndx`: Section header string table index.
- Decompile: FUN_00400600 - (packedup):** Shows the decompiled C code for the function. It is a loop that reads data from a memory location and writes it to a buffer.
- Console - Scripting:** Shows the output of the script, including the address `00400600`.



The vulnerability






MADE IN ENGLAND



Key: 012345



012345





Key: 01234567



012345	67
--------	----





Exploitation

General exploitation steps

1. Memory shaping
2. Create an information leak
3. Hijacking the control flow
4. Obtaining remote code execution



Memory shaping





Information leak


```

17 string sInput;
18 int iLength, iN;
19 double dblTemp;
20 bool again = true;
21
22 while (again) {
23     iN = -1;
24     again = false;
25     getline(cin, sInput);
26     system("cls");
27     stringstream(sInput) >> dblTemp;
28     iLength = sInput.length();
29     if (iLength < 4) {
30         again = true;
31         continue;
32     } else if (sInput[iLength - 3] != '.') {
33         again = true;
34         continue;
35     } while (++iN < iLength) {
36         if (isdigit(sInput[iN])) {
37             continue;
38         } else if (iN == (iLength - 3) )
39             continue;
40     }
41 }

```

DEP

Vollet Davis

Sofi Kaiser

Mike Nolan

Q3 Outlook

Design functional products to help people get out and experience the world

OFFER NEW SERVICES

We're giving 1-month trials for consumers to test out our new products.

INCREASED SUPPLY

Raising our backstock 40% in Q3 to meet Q4 Demands

OFFER 24/7 SUPPORT

Headcount increased – 20 new openings

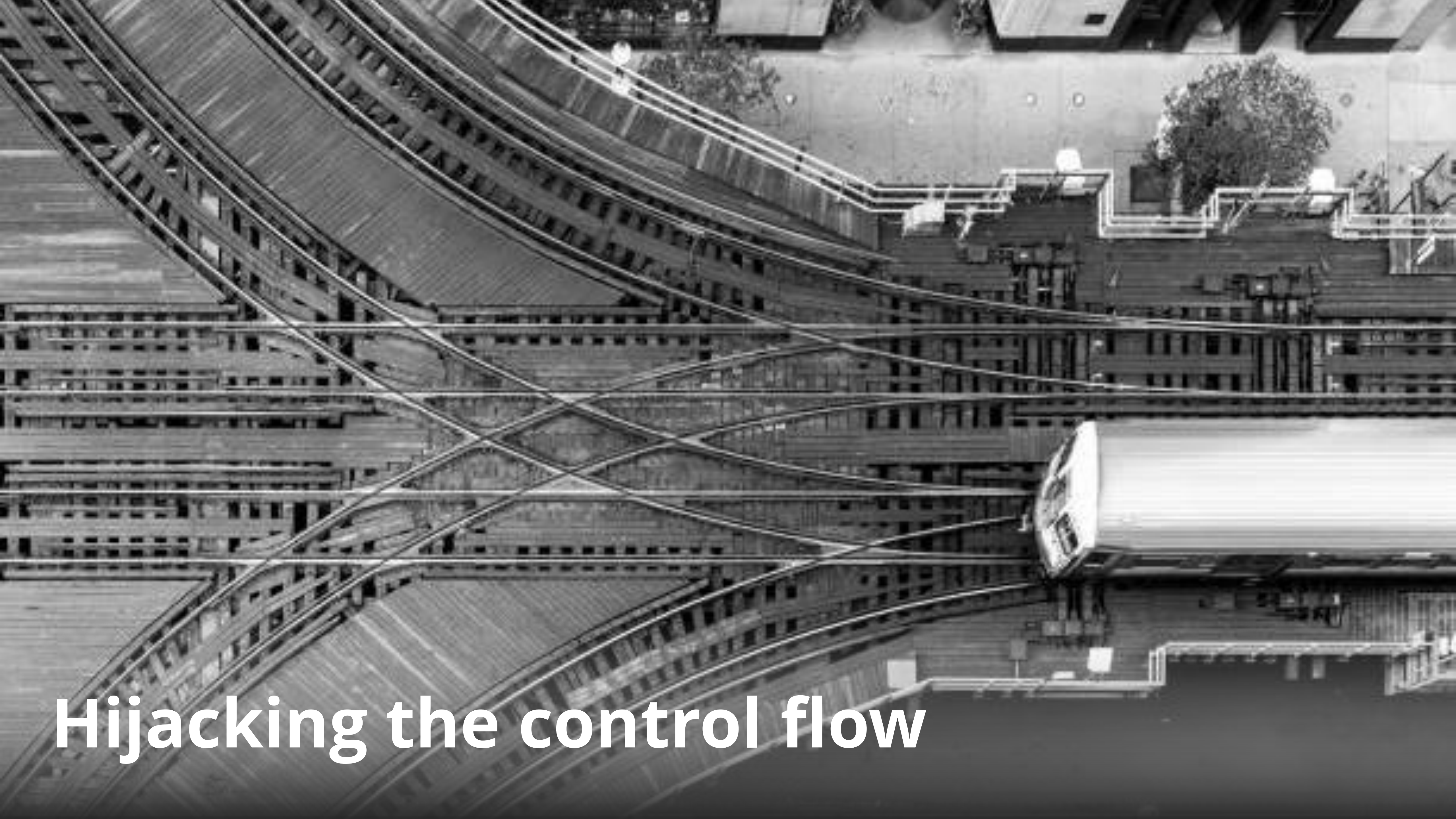
2 of 10

ASLR





Information leak



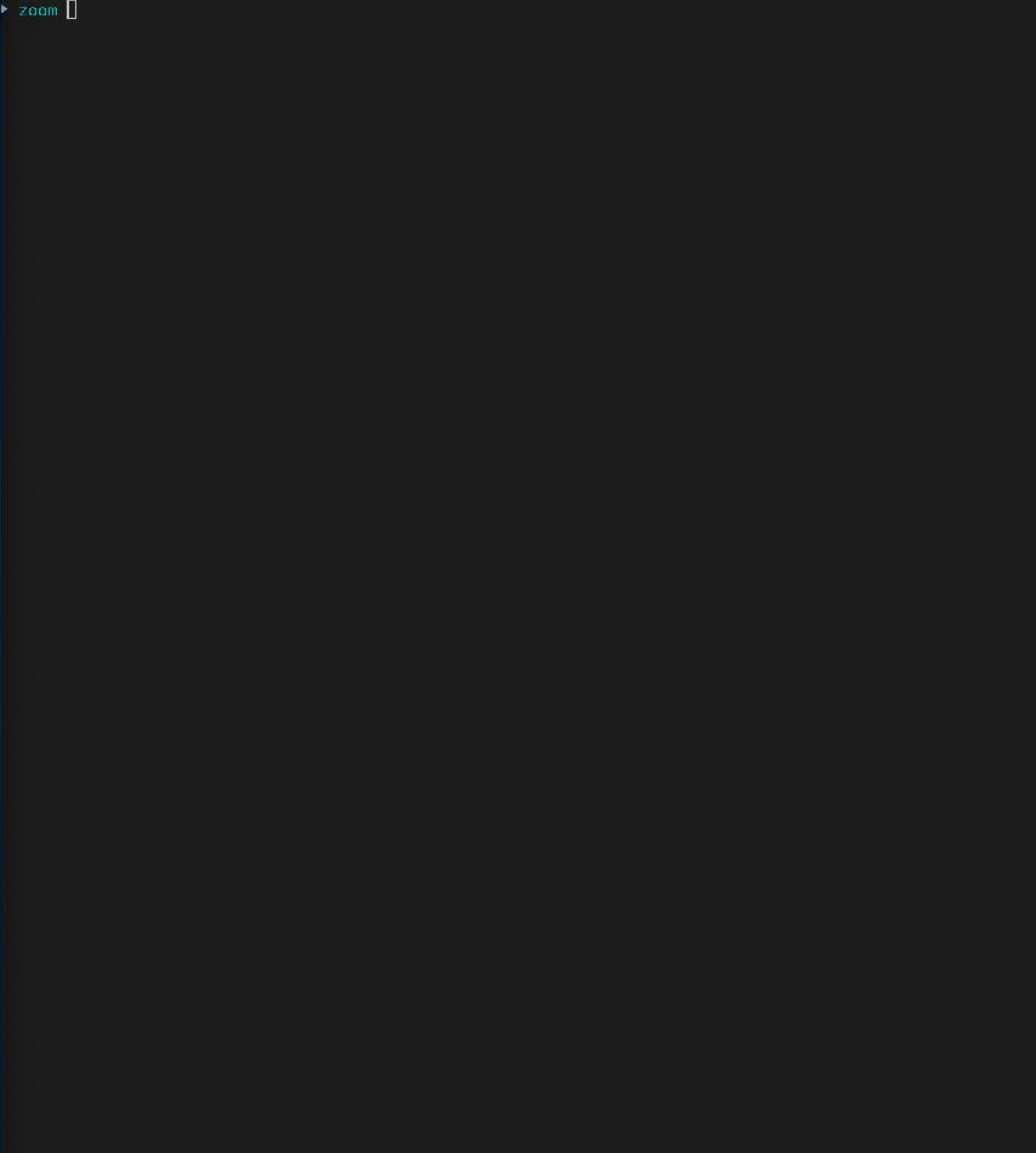
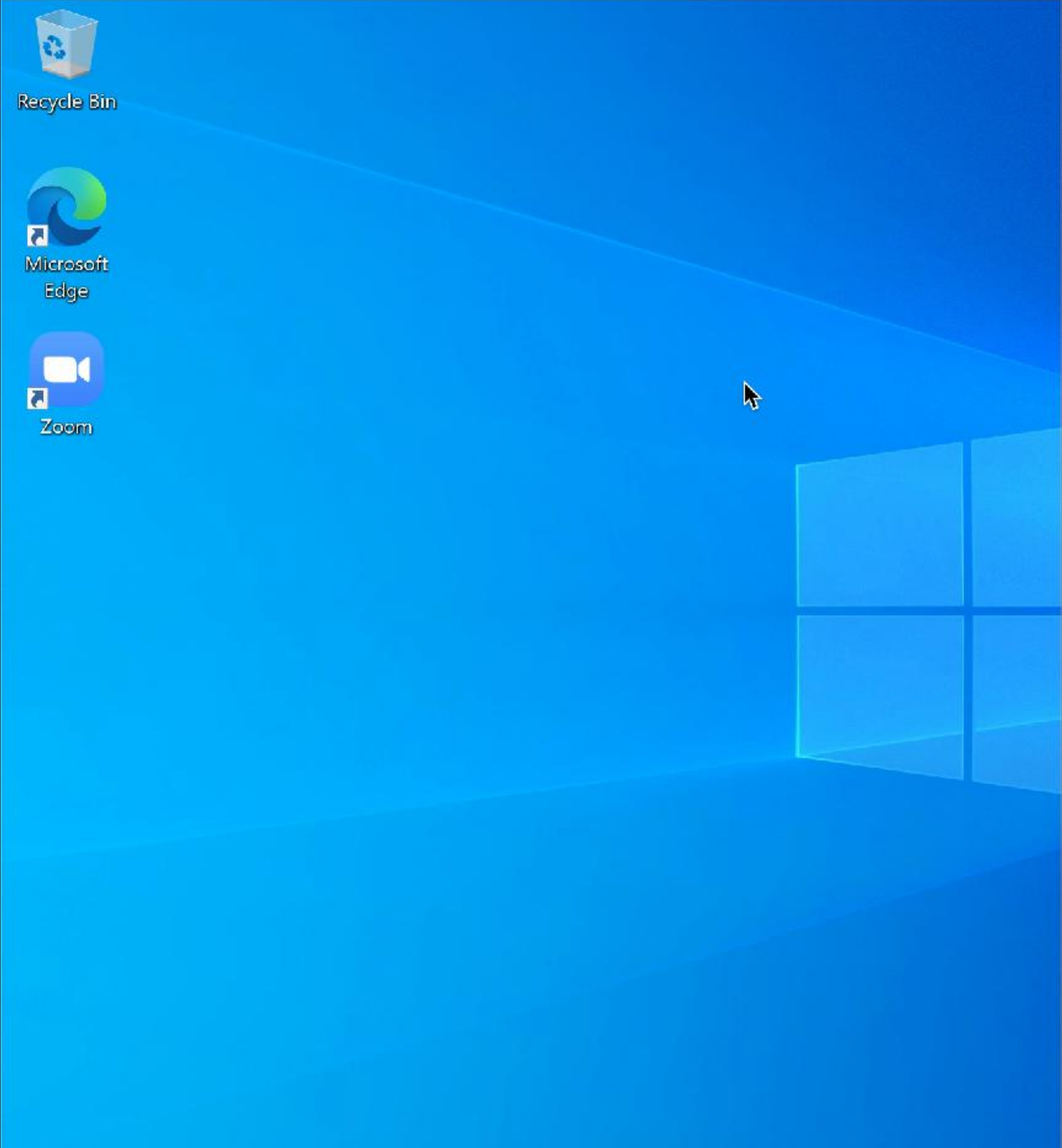
Hijacking the control flow

A close-up, low-angle shot of a hand holding a white remote control. The remote is held vertically, with the thumb resting on the top. The background is dark and out of focus, showing a television screen with vibrant, colorful bokeh lights in shades of blue, purple, and pink. The overall mood is dim and focused on the remote control.

Obtaining remote code execution



ROP chain

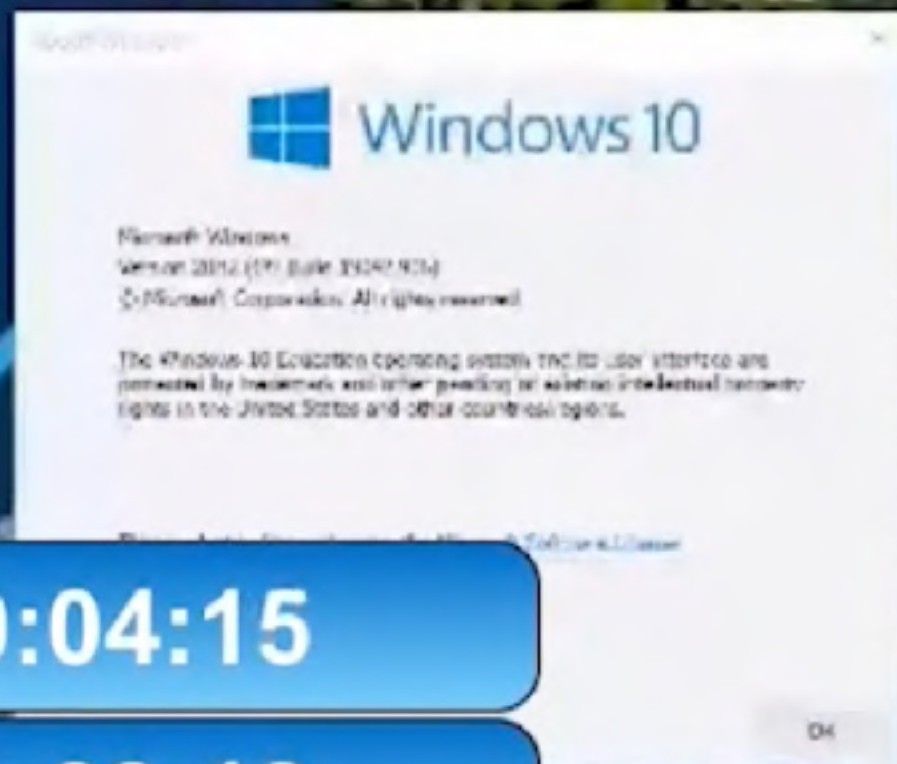
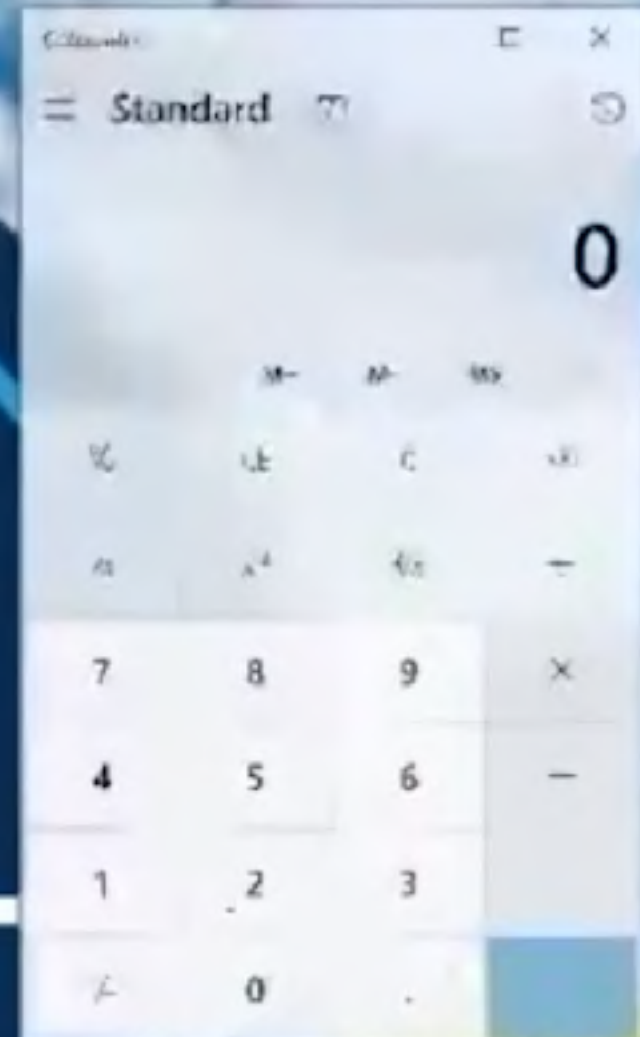
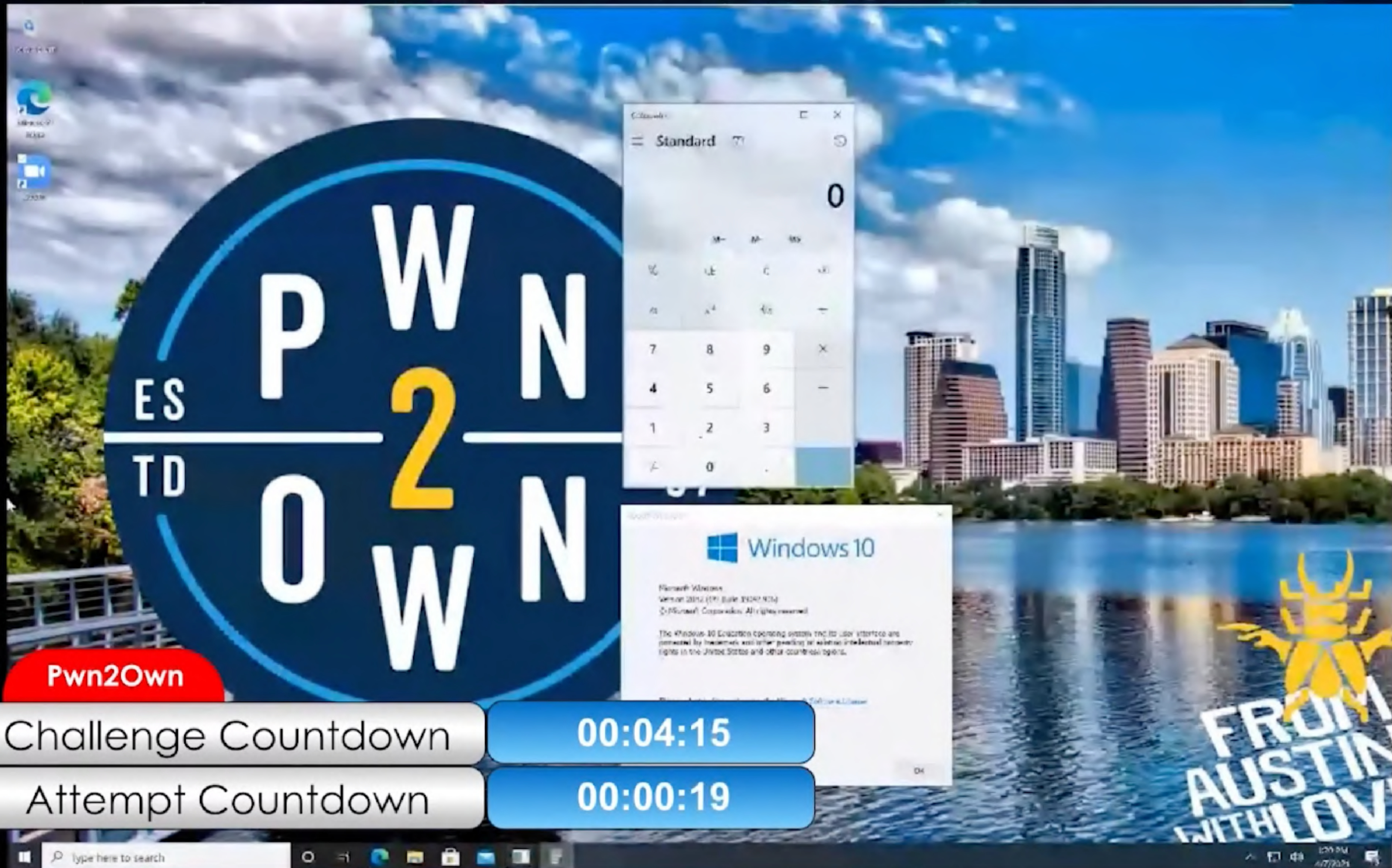




**Daan Keuper
Thijs Alkemade
Computest**



COMPUTEST



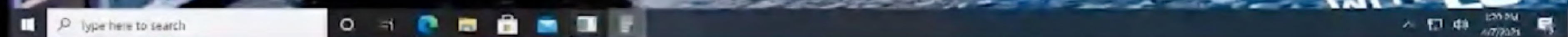
Pwn2Own

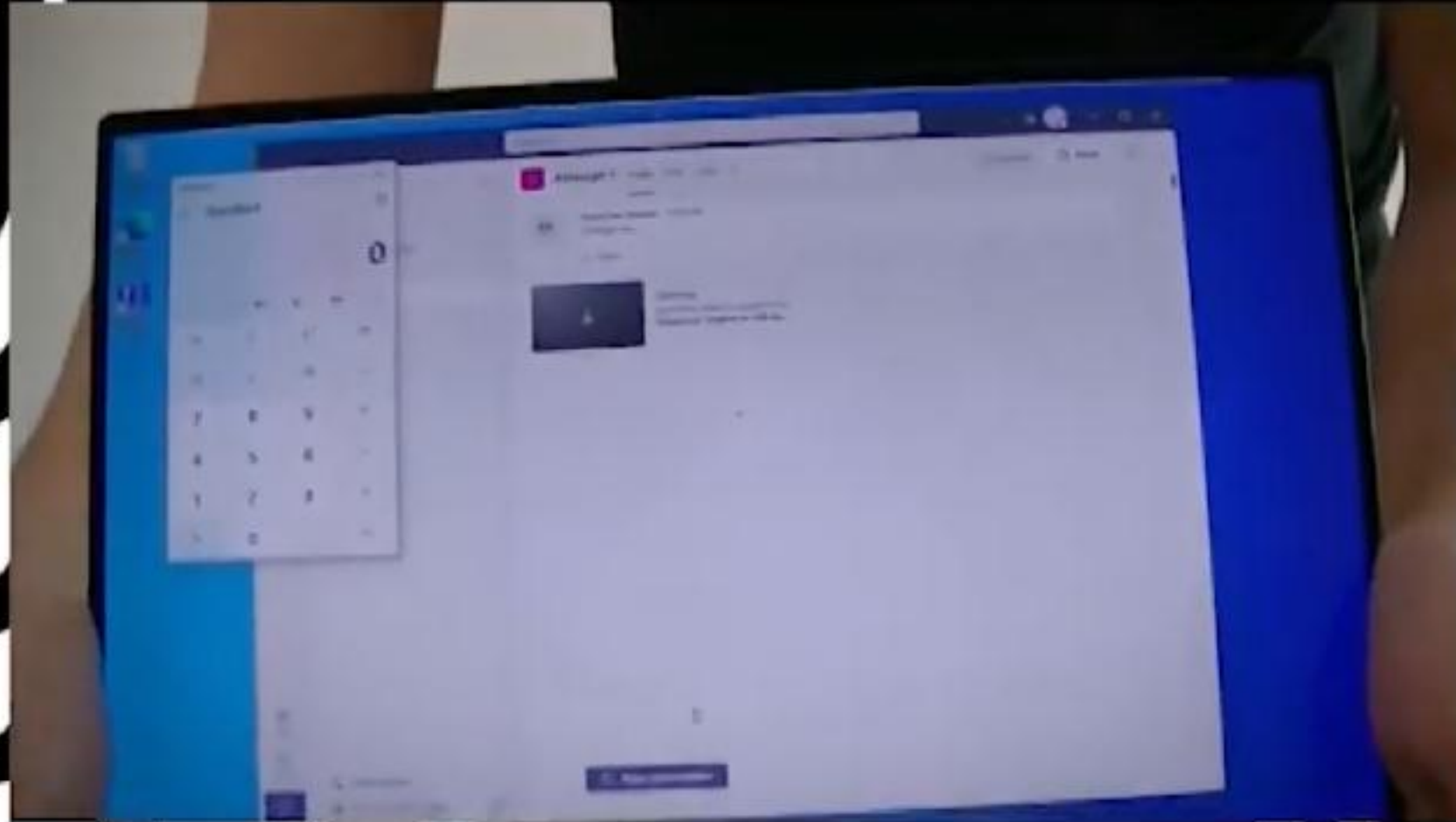
Challenge Countdown

00:04:15

Attempt Countdown

00:00:19





P



07

OV

Pwn2Own

Challenge Countdown

00:19:32

Attempt Countdown

00:04:31

Lessons





- > 1.5 weeks of reverse engineering for finding an exploitable vulnerability
- > 1.5 month of turning this vulnerability into a reliable exploit
- > Companies should keep in mind that these types of vulnerabilities can be in everything
- > Mitigations can increase time by a lot, but not perfect

```

movaps ptr [rbp+var_B0+10h], xmmword ptr [rbp+var_B0+20h]
movaps ptr [rbp+var_B0+30h], xmmword ptr [rbp+var_B0+40h]
test    ptr [rbp+var_B0+50h], xmmword ptr [rbp+var_B0+60h]
jz      ptr [rbp+var_B0+70h]
mov     esi, esi
mov     loc_100006F06, rdi
mov     rbx, rdi
mov     r15, esi
mov     r14d, r14d
mov     r13, [rbp+var_B0]

lea     rdi, [rbp+var_B4]
mov     rsi, rbx
mov     rcx, r15
mov     rcx, r13
call    _mbrtowc
cmp     rax, 0FFFFFFFFFFFFFFFh
jz      loc_100006F0B
mov     r12, rax
test    rax, rax
jz      loc_100006F15
mov     r12, 0FFFFFFFFFFFFFFFh
short  loc_100006EC5
edi, dword ptr [rbp+var_B4]
rdi, 7Fh
short  loc_100006EAA0
rax, cs: DefaultRuneLocale_ptr
eax, [rax+rdi*4+3Ch]
ecx, 40000h
eax, ecx
short  loc_100006EAA

```




<https://sector7.computest.nl/post/2021-08-zoom/>

Thank you for listening



research@computest.nl